Collaborative Project
Grant Agreement Number: 318240

Deliverable number:	D3.5	**Work package number:**	WP3
Due date of deliverable:	31.03.2014 (M18)	**Actual submission date:**	15.04.2014
Start date of the project:	01.10.2012 (M01)	**Duration:**	48 months
Nature:	Report	**Dissemination level:**	PP

Lead beneficiary:	PhoeniX Software
Contact person:	Arjen Bakker
Address:	Hengelosestraat 705, 7522 PA Enschede, Netherlands
Phone:	+31 53 483 6460
Email:	arjen.bakker@phoenixbv.com
Author(s):	Twan Korthorst, Remco Stoffer, Arjen Bakker
Contributing beneficiaries:	

Project Information

PROJECT

Project name:	Photonics for High-Performance, Low-Cost & Low-Energy Data Centers, High Performance Computing Systems: Terabit/s Optical Interconnect Technologies for On-Board, Board-to-Board, Rack-to-Rack data links
Project acronym:	PhoxTroT
Project start date:	01.10.2012
Project duration:	48 months
Contract number:	318240
Project coordinator:	Dr. Tolga Tekin – Fraunhofer IZM
Instrument:	Large-scale integrating project - CP-IP
Activity:	ICT-8-3.5 - Core and disruptive photonic technologies

DOCUMENT

Document title:	Physical layer model extraction towards sub-system layer
Document nature:	Report
Deliverable number:	D3.5
Due date of delivery:	31 March 2014 (M18)
Calendar date of delivery:	15.04.2014
Editor:	Twan Korthorst
Author(s):	Remco Stoffer, Arjen Bakker, Twan Korthorst
Lead beneficiary:	PhoeniX Software
Contributing beneficiaries:	
Dissemination level:	Consortium Confidential
Work package number:	WP3
Work package title:	Architectures, Protocols & Design of HPC and DataCenter Interconnect Systems
Date created:	11.03.2014
Updated:	28.04.2014
Version:	V1
Total number of pages:	8
Document status:	CO

PU = Public ; PP = Restricted to other programme participants (including the Commission Services) ; RE = Restricted to a group specified by the consortium (including the Commission Services) ; CO = Confidential, only for members of the consortium (including the Commission Services)

Table of Contents

1 Executive Summary

This deliverable describes the activities to include information from the fabrication processes and physical layer simulations into the design environment at sub-system layer. It will deal with methods to extract models and obtain information from fabrication databases.

Keywords: simulation, layout, process variations, circuit modelling

2 Integrated Product Creation Process

Successful research and development of micro and nano engineered products relies on an integrated design and manufacturing flow (i-PCP) incorporating product design, process development, fabrication and measurements (see figure 1). Contrary to the IC-industry the fabrication process (often referred to as "process flow") of the actual devices is a key parameter of the design. In a simplified way a designer in the IC industry has a two-dimensional design space (creating functionality by routing metal patterns), whilst a full three-dimensional design approach is required in the current status of the micro and nano engineering industry (creating functionality by etching fluidic channels, depositing optical waveguides, etc.). The creative process of the designer should be supported by a design methodology and corresponding tools providing the fabrication technology parameters (geometrical information and performance) integrated with the required physical simulations.

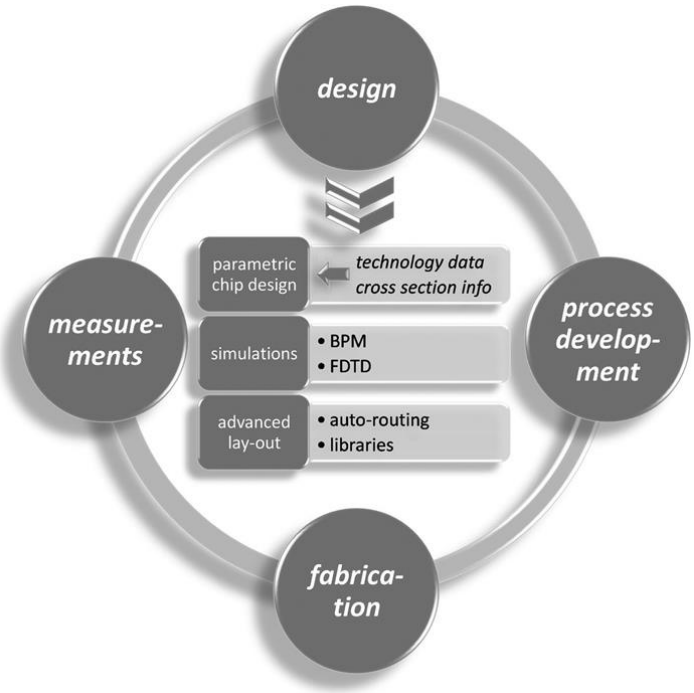

Figure 1: Schematic overview of the integrated product creation process

3 Design abstraction levels

Physically, there is a wide mix of technologies and functions in the envisioned PhoxTroT architecture. This varies from active components in III-V materials, to polymer and glass based motherboards or PCB's and fibers connecting racks. Al these different components have a different "position" in the functionality stack of the whole architecture.

For the design process, a different set of conceptual levels may be defined. The lower levels are relevant for the design of the chips, higher up the tree are the design methodologies for hybrid and composite circuits and the final system.
All these layers are incorporated in software tools and will need to create one integrated design flow. In deliverable D3.3 the PDAFlow API has been discussed and the aim of this API is to allow for tool interoperability and set a standard for defining the contents of design libraries (design kits or PDKs).

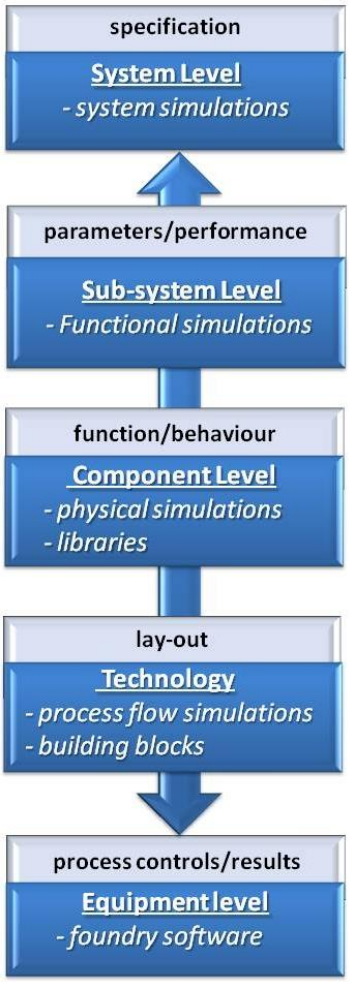

Figure 2. Conceptual design levels

In the design process, each level has its own set of simulation and physical description tools. Tools at one level receive information on desired functionality or settings from the next higher level, and use data from the next lower level to perform their tasks. Data passing from level to level is filtered to only provide the data that is actually needed by the receiving level. Figure 2 shows a subdivision into these abstraction levels. The blue boxes show the name of the level and the software tools that are needed in the level; the lighter boxes define the information that is passed up / down the chain.

4 Physical layer model extraction towards sub-system layer

To design/simulate at sub-system or circuit level, for instance in order to be able to create yield-optimized designs or proof a new concept, it is required that models are available to perform simulations without the need to run full physical (read electro-magnetic) simulations like FDTD or BPM. To create lower-computational-effort models of the components or building blocks on each level, data needs to be collected and passed to a higher layer in a more abstract way.

4.1 *Equipment Level*

On the equipment level, foundry software (database software for process flow design / control and data acquisition / management) provides, over time, the statistical information that is needed to estimate the process tolerances on the relevant parameters that are passed to the Technology level. Examples of quantities of which the tolerances will be generated are line width, etch speed, deposition speed, refractive index, alignment accuracy and doping concentration levels.

4.2 *Technology Level*

The technology level obtains tolerance data from the Equipment level, and processes this into variations in the cross-sections of the building blocks, using process flow simulations. This provides the tolerances on quantities like under-etch, etch depth and layer thickness to the Component level. Additionally, some parameters that the Component level needs, are directly passed on from the Equipment level, such as refractive index variances.

4.3 *Component Level*

The Component level performs physical simulations on the building blocks that are defined in the Technology level, and combines these blocks into library elements (and physically simulates the optical or electrical response of these library elements). By 'optical or electrical response' we mean how the amplitudes at the defined output ports of a Component are influenced by the amplitudes at the input ports. From the tolerances on the quantities that are passed from the Technology level, tolerances on all responses are calculated and passed to the Sub/system level.

4.4 *Sub-system Level*

On the subsystem level, functional simulations calculate the response of composite systems, typically using an S-matrix approach. The tolerances on the responses from the Component level translate into tolerances on all elements of the scattering matrix, from which the tolerance of the response of the sub-system can be calculated. The response of the subsystem is translated into performance parameters, which are passed to the System level.

4.5 *System Level*

On the system level, the performance data from the subsystems is used to simulate the behavior of the device in a real-world network application. This process extracts quantities like for example the bit error rate. Spreads on the performance parameters of the subsystems will directly affect the system-level performance. Comparing these numbers with the specifications will give a direct yield number.

5 Physical layer model extraction

The definition of a more abstract model based on physical parameters can be done by creating parametric sub-system models defined in the design kit per building block. These predefined models can adopt themselves based on externally provided parameters, like from a process or measurement database and/or from physical simulations.

As an example to validate our efforts in this task we take an Arrayed Waveguide Grating (AWG) wavelength (de-)multiplexer, as this is a widely applied component in integrated optics designs. The same methodology will apply for other composite building blocks.

An AWG can be considered to lie in the sub-system level being composed of an input section (the star coupler), a set of grating branches and an output star coupler, which can be considered to lie in the component level. To simulate a full AWG design with BPM will take too much computational power (read time), to be usefull when designing a circuit containing several more complex building blocks and investigating trade-offs between architectures, calculating process tolerances etcetera.

Since technological data of the variations in the four main technological parameters is available, it is possible to create simpler, interpolated models of the only important parameters of the building blocks of the AWG, by using mode solvers or Beam Propagation Methods. Using these models, the specs of the AWG – being quantities like central wavelength, wavelength spacing, 3 dB and 1 dB bandwidth, and cross-talk – can be determined for a given set of technological parameters. By subsequently analysing these specs for a large number of sets of parameters, one can get an idea of the yields of the device, and further analysis allows to determine which of the technological parameters has the highest impact on the performance of the device – thus giving information on which parameter the focus of technological improvement should be.

The four parameters of the structure at hand are the waveguide width and height and the refractive index of the core and the cladding (the substrate is assumed to be constant).

5.1 *AWG Simulation*

An Arrayed Waveguide Grating looks like shown below. Inside the grey squares, the inputs and outputs and the star couplers are located; they are connected by the red (and part of the blue) waveguides.

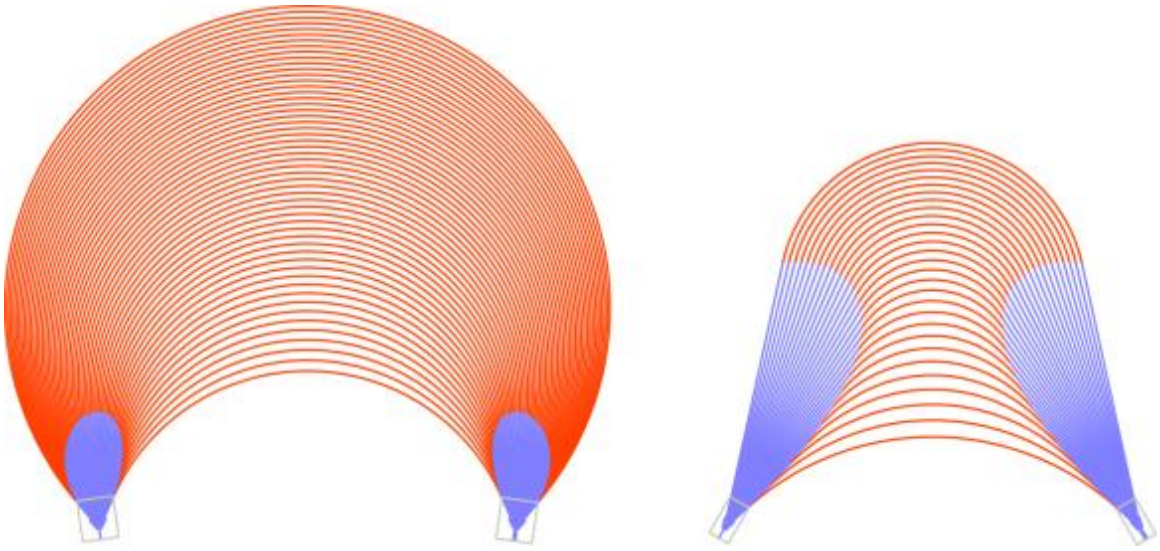

Figure 3: Impressions of an Arrayed Waveguide Grating.

The devices work by introducing a wavelength-dependent tilt on the phase front, thus focusing light of different wavelengths on different output ports. At the central wavelength, the optical length difference between two subsequent branches is an integer number times the wavelength, thus giving a flat phase front and focusing in the middle; higher and lower wavelengths focus on different output channels.

As said, an AWG is too large to simulate directly with physical simulation routines. The way to tackle this is by subdividing it: First, the first star coupler is simulated using Beam Propagation Methods (BPM), until the point where the array waveguide are decoupled. From there, the behaviour of the array waveguides can be approximated very well by a calculation of the propagation constant of a mode of the guides. The accumulated phase of the array guides is added to the phase of the modes at the output of the first star coupler and those fields are launched into the array branches of the second star coupler, which is subsequently BPM'ed. This approach, however, is rather slow; for each wavelength to be analyzed it takes multiple minutes, even for a 2D simulation.

5.2 *Scattering matrices*

Therefore, it is required to use a simpler model and we use a scattering matrix approach combined with interpolations. A simplified S-matrix of the star couplers and of each array branch is calculated.

To create the full scattering matrix of a star coupler, the response of the structure to input into each port (so each array branch and each input/output guide) should be calculated, and the amplitude and phase of the output fields at each output port recorded. However, we neglect reflections, and moreover, due to reciprocity the scattering matrix is symmetric; if ports of an optical circuit are isolated (like they are at the end of the star coupler BPM section when the waveguides have decoupled), the amplitude and phase of the transmission from port A to port B is the same as from port B to port A. This means that once we know the transmission from each input waveguide to all array branches, the entire scattering matrix is known. Furthermore, the (unwrapped) phase and the amplitude of the transmissions can, after minor modifications for the length difference of the branches between the end of the free-space section and the end of the BPM calculation window, reasonably be interpolated by a quadratic fit if simulations are done only at the central and the two outer inputs. Note that for the simulations of these inputs, the wavelength that is expected to be focused onto just those channels is used.

To obtain the scattering matrix of an array branch, one only needs to know its length (which is fixed for a given array layout) and its effective index. Since the wavelength range and the variations of the technological parameters are small, the effective index can be approximated using a first order Taylor expansion around the base (design) values. This is done using a Film Mode Matching mode solver.

In this way a model can be constructed for a complete AWG, that can be applied in circuit design software tools. The same methodoly applies for other composite building blocks and thus requires knowledge about the building block and how to properly model such a building block at a higher abstraction level. If this model has been constructed, the next step is define this in a standard, but parametrised, way into the design kit. Through the PDAFlow standard (see D3.3), a model that has been created and put into a design kit can be used by different software tools. Today, this is OptoDesigner and Aspic from PhoeniX Software and Filarete. Since recently also Lumerical and VPI Photonics are member of the PDAFlow Foundation and their sub-system simulation tools will be able to use these models in the same way as described below.

5.2.1 Example for process variations tolerance analysis

The AWG that will be used in the examples below is a 25 channel, 300-branch AWG with a channel separation of 0.4 nm, at a central wavelength of 1.55 mu in a "low-contrast" technology. The parameters, are given below.

	Base	σ
Waveguide Width	4.3 μm	78 nm
Waveguide Height	4.3 μm	110 nm
Core Index	1.466472	0.000284
Cladding Index	1.4448	0.000284

We assume in this section that the variations are uniform across the device, which may not be true – for an example of nonuniform variation analysis, see the next section.

5.2.2 Spectra

At the base values of the technological parameters, the calculated spectrum of the AWG looks as shown below:

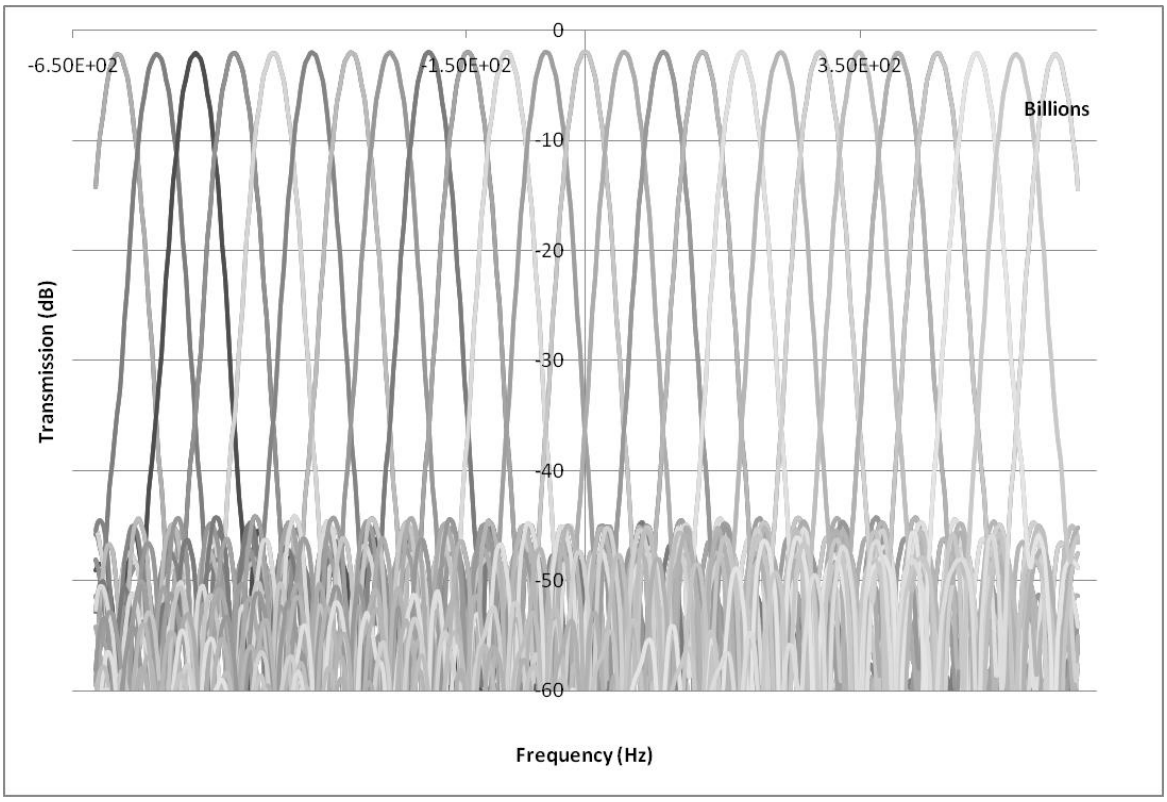

Figure 4: Spectrum of the ideal, 'base' AWG

The spectrum as shown is nicely centred around the central wavelength, with approximately 50 GHz spacing. Note that the graph shows both TE and TM – but since they are so closely matched due to the square waveguide geometry, they are near-indistinguishable.

When using Monte-Carlo setting a random variation, a spectrum that one may obtain is the following:

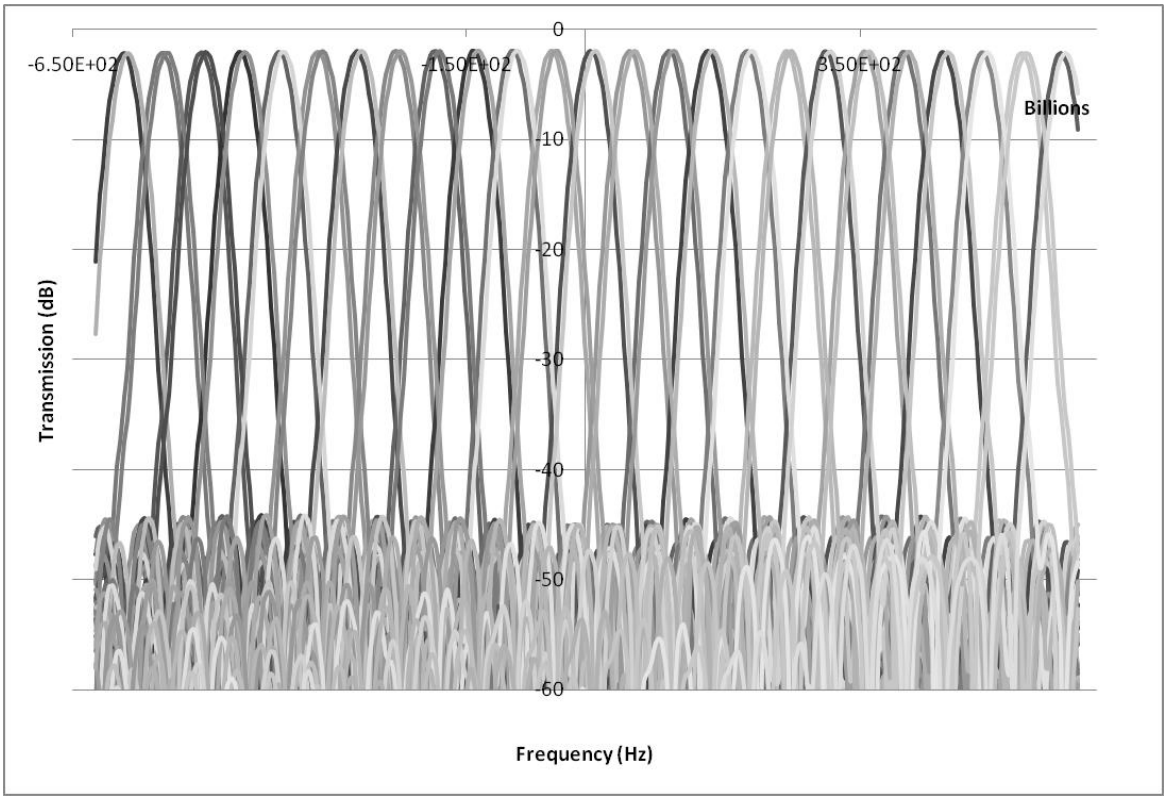

Figure 5: Spectrum of an AWG with n_core = 1.46619, n_cladding = 1.44467, width = 4.413 μm, and height = 4.247 μm

This spectrum is clearly shifted from the centre, and the TE and TM curves start to drift apart.

5.2.3 Full Monte-Carlo

On this design, we perform a Monte-Carlo analysis with 100 runs. We analyze the peak wavelength and loss of channel 1 (which should be at a wavelength 4.8 nm above the centre wavelength) and channel 13 (which should be the centre wavelength), the 3 dB and 1 dB bandwidth of channel 13, and the adjacent channel crosstalk of channel 13.

The figure below show the histograms of the wavelength difference from the central wavelengths of the two channels considered. They are both more or less centred around the proper value, but have a rather large spread; their standard deviations are both around 0.35 nm – which is almost a whole channel.

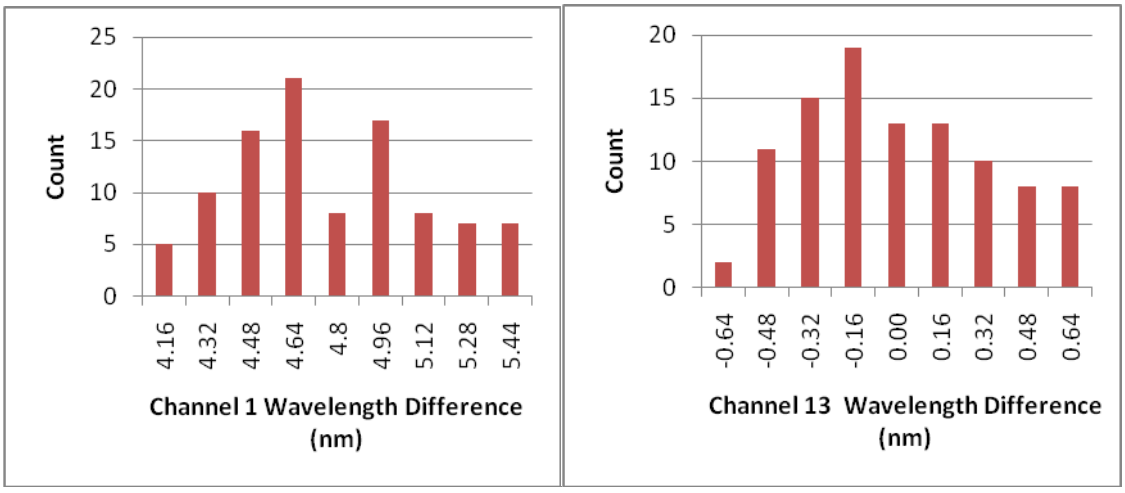

Figure 6: Peak positions

The bandwidth and crosstalk histograms show similar features, although the spread in these is too small to be of influence:

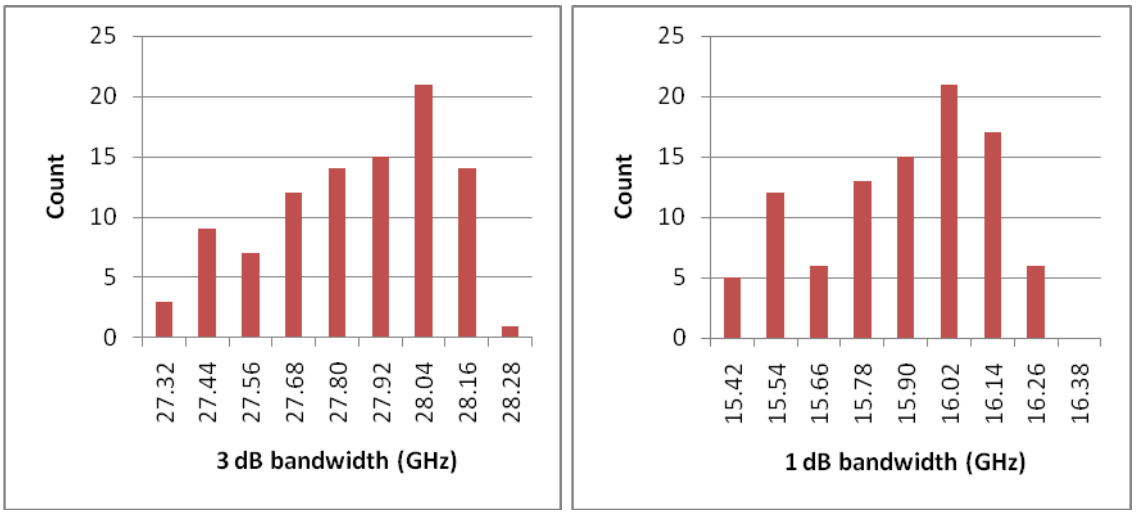

Figure 7: Bandwidth histograms

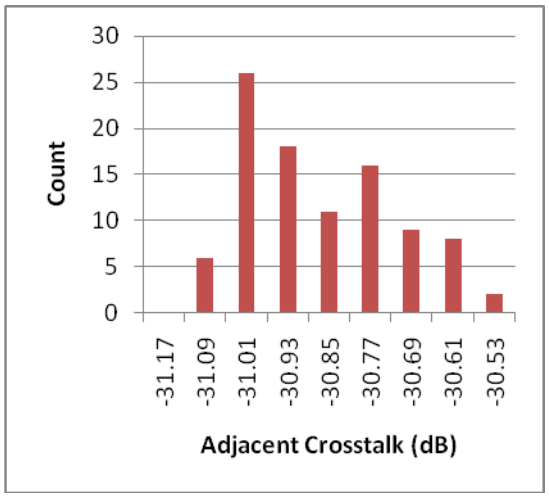

Figure 8: Adjacent crosstalk for the full Monte-Carlo analysis

The spread in the insertion loss is also very small – and is better visualized in a graph showing the IL versus the run number:

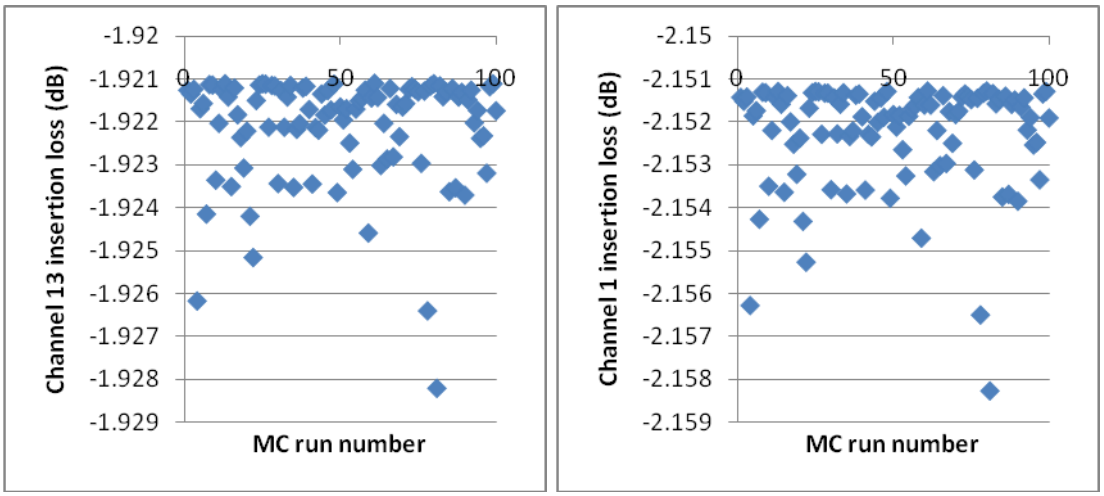

Figure 9: Insertion loss data for the two channels

5.2.4 Most Critical Parameter

In order to find the most critical technological parameter, the best way is by doing more Monte-Carlo analyses, while keeping one parameter fixed at its base value while the others are left free. The parameter for which the setting to zero causes the greatest improvement in quantities like yield or general variations is the one we are looking for. Since the wavelength shift of the channels is the result that varies most wildly due to the technology variations, we will look at the standard deviation of channel 13. The following table gives this standard deviation for the full MC, and when keeping one parameter constant:

Parameter Kept Constant	Standard deviation position channel 13
None	0.353 nm
Waveguide Width	0.328 nm
Core Index	0.182 nm
Cladding Index	0.322 nm
Waveguide Height	0.267 nm

Clearly, the core refractive index is the most critical parameter.

5.2.5 Nonuniform Array Branches

If the distribution of the technological parameters is not constant over the array branches, the spectrum may also be distorted. This effect is shown below, where the base value is used for the parameters, and each array branch is modified by a variation with a standard deviation that is only 2% of the 'normal' standard deviation. Clearly, this raises the noise floor a lot, and also increases the insertion loss.

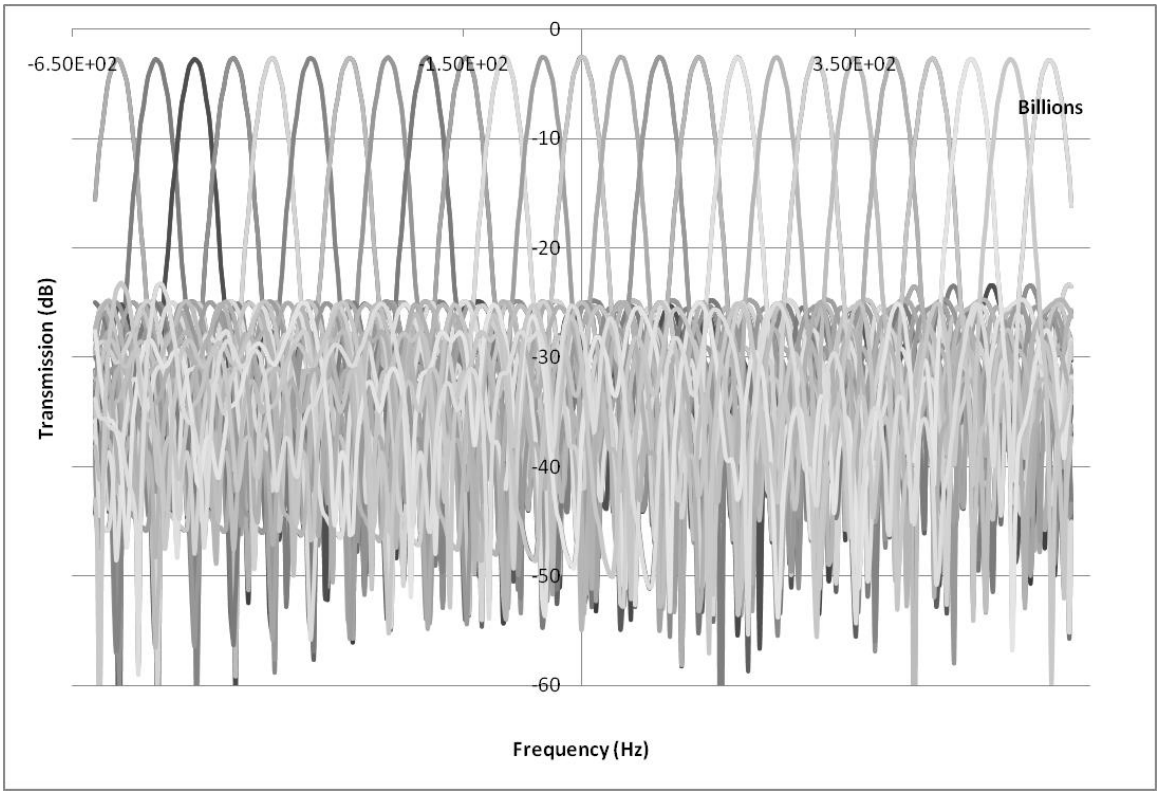

Figure 10: Spectrum of an AWG with variable parameters over the array branches

The presented methodology tool is capable of performing tolerance analysis on components as complicated as an Arrayed Waveguide Grating. It employs techniques to transfer information between the conceptual design levels, thus enabling technological variations to be translated into the performance parameter variations of the component. It can be used,

for example, to find the most critical parameter in fabrication, thus the one that needs the most attention when fabricating.

To be able to include process information and/or measurement data into the models as described in the design kit, it is needed to collect measurement data from the fabrication process, wafer tests, bare and packaged die.

6 Process and/or measurement data

To collect information from a manufacturing environment, commercial software tools are available in the market. These are called Manufacturing Execution Systems. This MES software has two purposes: to manage the fabrication processes and to collect performance information from the processes and the manufactured product(s). For the IC-manufacturing industry a few vendors exist and they have very sophisticated and advanced tools to obtain maximum efficiency and quality in a 24/7 manufacturing environment. For the micro and nano technology marketplace, including integrated photonics, PhoeniX Software is actively developing a dedicated solution, the Living Database. Major difference with existing IC-manufacturing solutions is the flexibility and the focus on the R&D phase of process and product development as well.

6.1 *Simlink: connection between Living Database and design tools*

Within the design tools and the design kit framework new functionality has been created to query data directly from The Living Database or any other MySQL database and use this in the lay-out and/or simulations. Furthermore it has been made possible to override the encrypted and compiled measurement data or model contents of a design kit (PDK) (see D3.3), with new/updated information. This is important to be able to receive and use last minutes updates from the suppliers of PDKs (including the foundry information, IP-blocks and packaging templates). Focus has been on the measurement data from the foundries, as this is the most likely scenario in practice.

PhoeniX Software makes use of a domain specific scripting language in it's software environment. This java/c like language can be used through the graphical user interface that will generate the underlying script lines and by entering the script directly into a script editor, that is part of the software. To accomodate new functionality, new script functions will have to be defined and the script parser will have to be extended to be able to interprete the new functions and execute those in the C++ environment. Below a few examples of the new script functions are depicted to illustrate this new functionality.

```
dsp::clearInfoWin();
string SqlLogin=/*server,database,user,passwd*/ "localhost/Bright/bright/bp";
/** Any SQL form.
 * This is handy for table joins or complex stuff. Multiple SQL columns will give
 * second []. The last is the format, which is: [a]{i|d|s}[a] where the [a] is an
 * optional specification that the value gives the axis; if at the left, then use
 * the first column. If at the right, then use last column.
 * i|d|s is the data type: integer, double or string.
 */
double itemA[][]=MySQL("sql:"+SqlLogin+":SELECT a,a+1,a*a FROM dbldata:d");
/** Table form.
 * Uses :<table>:<field>:[<selection>]:format
 * This is easier to read, since you do not have the SQL overhead of the SELECT's
 * etc.
 */
double itemB[]=MySQL("db:"+SqlLogin+":dbldata:b::d");

var item=itemA+itemB; // weird value...
item{"a"}=itemA;
item{"b"}=itemB;
item{"maxval"}=max(itemA,itemB);
item{"minval"}=min(itemA,itemB);
item{"b2"}=sqr(itemB);
res::SaveItem("/tmp/x.result",item);
```

Figure 11: Screenshot of script Editor with example code to describe the data in The MySQL database and get in into the PDK.

```
/** Until here it is the "getting data" -> dll.
 * From here it is getting it back into spt, but on a remote site (non-SQL access).
 * Configure in PDAConfig.ini using:
PHOENIX/RESULT/API2 = /tmp/y.result
 * to have updated *.result. From DLL the call would be:
 */

var itemRd=res::ReadItem("pdablob#<myfiletag>","<tag>");

 /**
 * instead of the file IO below.
*/

var itemRd=res::ReadItem("/tmp/x.result","API2");
printf(itemRd,"\n");
```

Figure 12: Screenshot of the Script Editor with example code to obtain the the data from the the PDK (or an external file or MySQL database).